Customization of Alerting and Dashboards in Health Insights

Health Insights releases 3.x use InfluxDB, Kapacitor Stack for achieving alerting functionality in Health Insights KPIs

Steps for Customization

- Export a stock / custom KPI, will be used as starting points
- Edit the kapacitor TICK script and/or dashboards with changes
- Update KPI json with metadata for the changes above
- Re-Import the KPI into Crosswork, and test.
- Use this document in addition to the following **Health Insights Developer Guide**
- This is an iterative process until the desired result is achieved

https://developer.cisco.com/docs/crosswork/#!custom-kpis

Anatomy of a KPI

Refer to the KPIs made available on the Github for the stock kpis included as part of Crosswork relase.

https://github.com/CiscoDevNet/Crosswork-KPI

KPI consists of a json file that captures the `Metadata`, `Sensor list`, one or more `Alerting logic` (pointer to the TICK files) and one or more `Dashboards` for visualization of data and alerts.

Here is an example KPI for Memory utilization

`Metadata`

```
{
  "kpi_id"             : "pulse_memory_utilization",
  "kpi_name"           : "Memory utilization",
  "category"           : "Memory",
  "summary"            : "Monitors memory usage across route processor and line cards on routers",
  "details"            : "Monitors memory usage across route processor and line cards on routers; generates an aler
  "sensor_type"        : "YANG_MDT",
  "module_and_revision": "Cisco-IOS-XR-nto-misc-oper:2015-11-09"
}
```

`Sensor List`

Here we can see the settings related to default, minimum and max cadences that can be customized.

```
{
  "sensor_groups": {
          "sensor_group": [
            {
              "cadence": {
                "default": 120,
                "min": 10,
                "max": 900,
                "increment": 10
              },
              "sensor_paths": {
                "sensor_path": [
                  {
                    "path_id": "Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary/ram-memory"
                  },
                  {
                    "path_id": "Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary/free-physical-memory"
                  },
                  {
                    "path_id": "Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary/system-ram-memory"
                  },
                  {
                    "path_id": "Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary/free-application-memory"
                  }
                ]
              }
            }
          ]
        }
}
```

`Alerting Logic`

Each of the alerting script is based on a master template, in this case we are using stream based standard deviation based template. Main things to see is that scope of the script is local indicating limited to this KPI, and the `script_id` is the pointer to the TICK file that contains the logic that would get executed for every data point and it runs in Kapacitor docker container called `robot-astack-kapacitor`

```
{
  "scripts": {
    "script": [
      {
        "script_id": "pulse_memory_utilization_template.tick",
        "script": "",
        "template": "stream-stddev-activate-master-template",
        "template_vars": "memory_utilization_vars.json",
        "type": "stream",
        "scope": "local"
      }
    ]
  }
}
```

`Dashboards`

For this KPI we have 2 dashboards and will be shown in the device view page of Health Insights, and label indicated in this json file will be used as the `Label` in the UI. `value` here points to the actual `Grafana` dashboard json file. These Grafana dashboards will have a `InfluxQL` queries to show the right KPI data and alerts and uses template variables for runtime customization.

If this dashboard needs to be customized, we should have to updated the pointed json file or add a new one etc.

```
{
  "dashboards": {
    "dashboard": [
      {
        "type": "jsonfile",
        "value": "Pulse-memory-utilization-summary.json",
        "label": "Summary"
      },
      {
        "type": "jsonfile",
        "value": "Pulse-memory-utilization-raw.json",
        "label": "Raw"
      }
    ]
  }
}
```

Anatomy of Alerting

Alerting logic implemented as a TICK script is executed using `Kapacitor` on `robot-astack-kapacitor` docker container. Each KPI can have zero or more of these and are installed as `tasks` on kapacitor. These tasks are installed per KPI `profile`, `script`. And can be seen on kapacitor using `kapacitor list tasks` command. These tasks will have names following a scheme as follows…

`cw.pulse_<profile_id>_<script_id>`

Example: cw.pulse_ `core` _ `pulse_cpu_utilization`

Kapacitor can execute these tasks as `Stream` or `Batch` jobs and for the most part Health Insights will be using stream tasks.

TICK script can be represented as a `Flow Diagram`. Flow Diagram has nodes and the data stream flows one node to another node, each node does a specific action on the data stream. A data point in the stream has set of keys and timestamp identifying the point uniquely in time and from a specific network entity. And each point has as many values as the `InfluxDB` measurement this Tick script is processing the data from.

Here is the TICK Script for this Memory Usage KPI

Please refer to TICK script reference for additional details on the syntax and specification

Basic structure of the TICK script or the Flow Diagram are as follows…

- **A set of variables**
    - Some of these are exposed via UI and some are internal and some are fixed and should not be modified for customization
    - Some of these variables can be updated to have a certain default value and can be used for customization

    ```
    //KPI_INTERNAL KPI ID of the kpi this TICKscript measures
    var kpi_id = 'pulse_memory_utilization'

    //KPI_INTERNAL Alert ID for this KPI Alert event
    var alert_id = 'pulse_memory_utilization'

    //KPI_INTERNAL Alert message format string
    var alert_message = '{{ .Level }} : {{ index .Tags "node-name" }}  Memory Utilization: {{ index .Fields "kpi_

    //KPI_INTERNAL Clear message format string
    var clear_message = '{{ .Level }} : {{ index .Tags "node-name" }}  Memory Utilization: {{ index .Fields "kpi_

    //KPI_INTERNAL Measurement in influxDB to retrieve the data from
    var measurement = 'Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary'

    //KPI_INTERNAL Optional where filter expression to filter data stream before passing for transformation
    var where_filter = lambda: TRUE

    // KPI_INTERNAL filter var to filter data only related to kpi profile id
    var where_producer = lambda: strContains("ProfileIDs", profile_id)

    //KPI_INTERNAL Optional list of group by dimensions
    var groups = ['Producer', 'node-name']
    ```

```
//KPI_INTERNAL Lambda expression to reduce raw data to KPI
var stream_eval_lambda = lambda: (1.0 - "free-application-memory"/"system-ram-memory") * 100.0

//KPI_THRESHOLDS (Critical Alert Threshold) Generates an alert when the data sample does not fall within the
var crit_threshold = 2.0

//KPI_THRESHOLDS (Warning Alert Threshold) Generates an alert when the data sample does not fall within the r
var warn_threshold = 2.0

//KPI_THRESHOLDS (Clear Threshold) When a data sample falls within the number of standard deviations from the
var clear_threshold = 2.0

//KPI_THRESHOLDS (Activation Threshold) Values must exceed the activation threshold to generate a standard de
var activation_threshold = -1.0

//PULSE_RUN_TIME (Downsample Factor) Down sample factor for alert evaluation applied on data stream (Default
var downsample_factor = 1
```

`kpi_id`, `alert_id` is something you may not want to change as they need to match the kpi_id from kpi json file, else it will break some of the functionality.

- **Input Data Query, Filter, Grouping and Down Sampling**
  - This section of the Flow Diagram shows how input data stream is defined
  - From measurement, filter out any data point that is not be processed for Alerting
  - Group the series by a set of keys so that alerting is for a given uniq network element
    - Example: group-by `node-name` will do memory utilization per `RP/Linecard`
    - Note that the keys and their names are part of the `YANG` or `MIB` model itself and these are called as `tags` in `Influx` terminology. Refer to the model for the keys that would be applicable for a certain `xpath`
  - Customization here can be done by changing the variables `where_filter` and `groups`

```
var data = stream //stream0
|from() //from1
    .measurement(measurement)
    .where(where_producer)
    .where(where_filter)
    .groupBy(groups)
    .quiet()
|sample(downsample_factor) //sample2
```

`lambda` is a keyword used to indicate mathematical or logical expression and is a short for a function call.

We are using `lambda` expression to filter the data stream, and in this example we are saying consider only data points where `system-ram-memory` is > 8GB. Same goes with `groups` which is a list of keys and one has to be careful as if fewer keys are used then we need to use aggregation functions in next Flow Diagram nodes.

`Downsample` is exposed via the UI and not needed in any customizations as it can be changed while creating `KPI Profile`

```
//KPI_INTERNAL Optional where filter expression to filter data stream before passing for transformation
var where_filter = lambda: "system-ram-memory" > 80000000

// KPI_INTERNAL filter var to filter data only related to kpi profile id
var where_producer = lambda: strContains("ProfileIDs", profile_id)

//KPI_INTERNAL Optional list of group by dimensions
var groups = ['Producer', 'node-name']
```

- **Transform to KPI Stream**
  - This section of the Flow Diagram shows how input data stream after it was filtering and grouped to multiple series, transformed to KPI Stream.

- The input points would contain a set of keys (`node-name` in this example) and these values, `ram-memory` , `free-physical-memory` , `system-ram-memory` , `free-application-memory`
- `lambda` expression is used to pick only select field or any other math computation
- `lambda` is a short expression instead of a full function(), please see Kapacitor Documenation
- To customize the kpi expression, modify the variable `stream_eval_lambda` , in this example its set to `lambda: (1.0 - "free-application-memory"/"system-ram-memory") * 100.0` to compute the percentage memory utilization
- **Note:** all the data types of the values used in the expression has be of the same data type. as a practice we are using float type for all numbers. Expression can also have string processing as its done in some of the stock KPIs
- From this node onwards only values available are the keys and `kpi_stream`

```
    // Node: transform
    // Input: data
    // Output: kpi_stream
    // Details: takes raw data and transforms it to kpi based on the stream_eval_lambda funtion
    var transform = data
    |eval(stream_eval_lambda) //eval3
    .as('kpi_stream')
    .quiet()
```

- **Alert Logic**
  - This section of the Flow Diagram has mainly three parts and varies from master template to template_vars
  - Please refer to the **Health Insights Developer Guide** for different types of logic that is exposed using KPI wizard
  - **Section 1**, to handle post processing of the `kpi_stream` before alerting comparisons like taking a first order derivative, standard deviation, summarizations like distinct count etc.
  - Here is an example of standard deviation based method, and this section for customization would mean that that you are building a new template and reach out to Development team for any guidance here.

```
// Node: stream_sd
// Input: transformed data
// Output: Sigma, alert_duration, clear_duration
// Details: Takes the transformed kpi data and identifies how many SDs away the sample is from mean
// And keeps track of how long the value is in alert or clear condition
  var stream_sd = transform
      |eval(lambda: sigma("kpi_stream")) //eval4
          .as('sigma')
          .keep('kpi_stream', 'sigma')
      |default() //insert the thresholds into the data stream //default5
          .field('crit_threshold', crit_threshold)
          .field('warn_threshold', warn_threshold)
          .field('clear_threshold', clear_threshold)
          .field('activation_threshold', activation_threshold)
          .tag('kpi_id', kpi_id)
          .field('alert_src', 'TICK')
          .tag('profile_id', profile_id)
```

  - **Section 2**, to handle the Critical/Warning check and write alerts, checks are made with expression `"sigma" >= warn_threshold` and `"sigma" >= crit_threshold`
  - `crit_threshold` and `warn_threshold` and other thresholds have some default values but are exposed using UI and can be changed during creation of `KPI Profile` . However, the default values can be customized here by changing variable assignments.

```
    //KPI_THRESHOLDS (Critical Alert Threshold) Generates an alert when the data sample does not fall within the nu
  var crit_threshold = 2.0

  //KPI_THRESHOLDS (Warning Alert Threshold) Generates an alert when the data sample does not fall within the numbe
  var warn_threshold = 2.0

  //KPI_THRESHOLDS (Clear Threshold) When a data sample falls within the number of standard deviations from the mea
  var clear_threshold = 2.0
```

- **Note** the `alert_message` used here which is a templated string in the list of variable we had in the very first section of this `TICKscript` which can be used to for customizing the message.

```
//KPI_INTERNAL Alert message format string
var alert_message = '{{ .Level }} : {{ index .Tags "node-name" }}  Memory Utilization: {{ index .Fields "kpi_stre

//KPI_INTERNAL Clear message format string
var clear_message = '{{ .Level }} : {{ index .Tags "node-name" }}  Memory Utilization: {{ index .Fields "kpi_stre
```

- Here is the snippet for the Alerting section

```
/ Alerting Node for warning and crtical
// Input: stream_sd
// Output: Warning and critical alerts
// TODO need to understand the exact format the ASTACK requires the alerts to be
// generated and posted
stream_sd
    |alert() //alert6
         // Warning on warning threshold
         .warn(lambda: ("sigma" >= warn_threshold AND (("kpi_stream" >= activation_threshold AND activation_thres
         // Critical on critical threshold
         .crit(lambda: ("sigma" >= crit_threshold AND (("kpi_stream" >= activation_threshold AND activation_thres
         .stateChangesOnly()
         .levelTag('level')
         .id(alert_id)
         .idField('id')
         .message(alert_message)
         .messageField('msg')
    |where(lambda: "level" != 'OK') //filter OKs out //where7
    |influxDBOut() //influxDBOut8
        .database(alerts_db)
        .retentionPolicy(alerts_rp)
        .measurement(alerts_meas)
        .tag('state', 'alert')
```

- **Section 3**, to handle the Clear condition and write alerts
- we can customize the `clear_message` to suite the need

```
// Alerting Node for clear condition
// Input: stream_sd
// Output: clear alerts
// TODO need to understand the exact format the ASTACK requires the alerts to be
// generated and posted
stream_sd
    |alert() //alert9
         //info on clear threshold
         .info(lambda: ("sigma" < clear_threshold))
         .stateChangesOnly()
         .levelTag('level')
         .id(alert_id)
         .idField('id')
         .message(clear_message)
         .messageField('msg')
    |where(lambda: "level" != 'OK') //filter OKs out //where10
    |influxDBOut() //influxDBOut11
         .database(alerts_db)
         .retentionPolicy(alerts_rp)
         .measurement(alerts_meas)
         .tag('state', 'clear')
```

- The stock kpis may have additional templates and this document is to explain how one can export and use it as a starting point for customization. Refer to https://developer.cisco.com/docs/crosswork/#custom-kpis/health-insights-stock-kpis

- **Best practices and other tips**

    - when exporting the stock kpis, `kpi_id` will have `pulse_` as prefix and is reserved for stock kpis and custom KPI would need to change this `kpi_id` not to use `pulse_` prefix
    - `kpi_id` value in kpi json is referenced in `TICKscript` and `Dashboards` remember to update it across all the files
    - `alert_id` value needs to have `kpi_id` as prefix in order for Health Insights to quickly query for all alerts related to the KPI
    - KPI json and Dashboard files validate that they are properly formed json files before testing

- **Debugging Changes to Alerting**

    i. After making changes to the alerting logic ensure you are validating the fallowing, .json files should be checked against a `json-lint` for any errors. Validate that any files referenced in `kpi` json are present
    ii. Import the KPI and fix any errors from Crosswork, Create a KPI profile and enable it on a device where this `kpi` would work.
    iii. Validate that KPI enable is successful, this would indicate that `TICK` script has not syntax issues and a `kapacitor` job is created
    iv. If the above step fails , check the errors in the job details or log into Crosswork and use `kapacitor` CLI command in `robot-astack-kapacitor` container for errors
    ```
    kapacitor list tasks
    tail /var/log/robot/kapacitor_stdout
    tail /var/log/robot/kapacitor_stderr
    ```